

## Client-Side Scripting

---

- So far, the browser has only passively displayed content.
- It is also possible to download a program and have it execute on the client browser
  - JavaScript / ECMAScript
  - Jscript
  - VBScript
  - TCL

## Browser as Development Platform

---

- Homogenous architecture
  - Single client abstraction
  - Write once, “run in many architectures”
- Easy to deploy and maintain
  - Download on demand
- “Efficient”
  - Client already has libraries
  - Code is small

## JavaScript

---

- a.k.a. ECMAScript
  - <http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM>
- An “object-based” programming language
- Syntax is very close to Java
- A web browser provides an ECMAScript host environment
  - Objects that represent windows, menus, pop-ups, dialog boxes, text areas, anchors, frames, history, cookies, and input/output.
- Attach scripting code to events
  - Focus change, page and image loading, unloading, error and abort, selection, form submission, and mouse actions.

## Relationship to Java

---

- Netscape originated *LiveScript*, later renamed to JavaScript at the last minute.
  - Invented by Brendan Eich at Netscape
  - Perceived competition with Sun Microsystems Java for client-side scripting
- Microsoft has a similar thing called JScript
- Complementary
  - JavaScript
    - Lacks multi threading, limited I/O
  - Java
    - Cannot interact with Browser or control content

## JavaScript Security

---

- Language/API limitations:
  - No file/directory access defined in the language
  - No raw network access. Limited to either
    - load URLs
    - send HTML form data to
      - web servers, CGI scripts, e-mail addresses
  - 'same origin policy'
    - can only read props of documents and windows from the same place: host, port, protocol
- Privacy restrictions:
  - cannot read history
  - cannot hide/show menubar, status line, scrollbars
  - cannot close a window not opened by itself

## Embedding in HTML

---

- Directly

```
<script type="text/javascript">
.....
</script>
```
- Indirect

```
<script type="text/javascript" src="test.js" />
```
- Location
  - <head> Definitions that are later called by elements in the document body
  - <body> Process while parsing

## Evaluation and Execution

- Evaluation
  - As document is parsed, in order
- Execution
  - Statement outside functions
    - When it is encountered
  - Statement inside function
    - When function is called
      - Event handler

```
<body onload="helloWorld()">
```

## Dynamic Typing

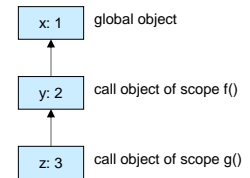
- Different than Java or C
- Variables can hold any type of value:
  - Number (8-byte IEEE fp)
  - Boolean
  - Function (first-class data type)
  - Object
  - Array (elements can be of mixed types)
  - String
- ... and can hold values of different types at different times during execution

## Variables

- Declaration
  - Explicit `var i = 12; // no type declaration`
  - Implicit `msg = "hello";`
- Scope
  - Global
    - Declared outside functions
    - Any variable implicitly defined
  - Local
    - Explicit declarations inside functions

## Scope Chain

```
var x=1;
function f() {
  var y = 2;
  function g()
  {
    var z = 3;
  }
}
```



## Functions

```
function functionName ([arg1] [...,argN])
{
  .....
  [return [value]];
}
```

- Arguments
  - Primitive types (string, number, boolean) are passed by value
  - Object types are passed by reference

## Control and Looping

- Control
  - if
  - switch
- Looping
  - for
  - while
  - do..while
  - for .. in
    - for (*property in object*) {}

## Objects

- JavaScript is Object Based rather than Object Oriented
  - No *class* construct
  - No inheritance, polymorphism.
- Objects are most like associative arrays

```
var point = new Object();
point.x = 2;
point["x"] = 2;           // equivalent to previous
point.print();
point["print"]();        // equivalent to previous
```
- Note: Object's definition is determined at *run time*.
  - Possible to dynamically add new properties or methods and to change the binding of methods at runtime

## Object Constructors

```
function Rectangle_area() {
    return this.width * this.height;
}
function Rectangle(w,h) {
    this.width = w;
    this.height = h;
    this.area = Rectangle_area;
}
var rec = new Rectangle(2,4);

rec.area();
rec["area"]();
//Rec.newFunction = newPredeclaredFunc;
```

## Object Constructors

```
function Rectangle(w,h) {
    this.width = w;
    this.height = h;
    this.area = function () {
        return this.width * this.height;
    };
}
var rec = new Rectangle(2,4);

rec.area();
rec["area"]();
// Rec.newFunction = newPredeclaredFunc;
```

## Predefined Objects

- Native/Built-in
  - Number
  - Boolean
  - String
  - Array
- Host
  - navigator
  - Window
  - Document

Primitive types are automatically coerced into Objects

## Array Object

```
var a = new Array();           // empty array
var b = new Array("dog", 3, 8.4);
var c = new Array(10);        // array of size 10
var d = [2, 5, 'a', 'b'];

c[15] = "hello";             // implicit extension
```

## Array Properties and Methods

- length
- join()
- reverse()
- sort()
- concat()
- slice()
- splice()
- push() / pop()
- shift() / unshift()
- toString()
- .....

## navigator Object

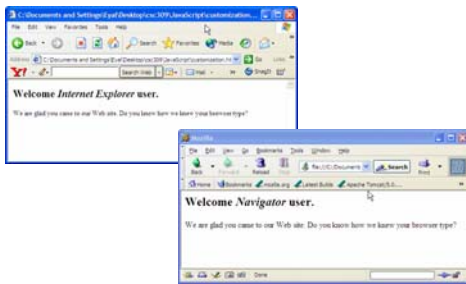
- Represents the Browser
- Properties
  - appName Browser name
  - appVersion Browser version
  - platform Client OS
  - cookieEnabled
  - cpuClass
  - Language
  - mimeTypes

## document Object

- Container for all HTML-related objects
  - Content nested inside <html> tag
- Methods
  - open opens output stream to document overwrites document
  - write appends text to document
- Properties
  - URL

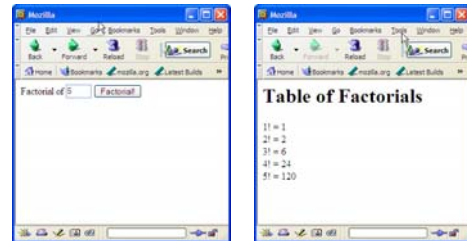
## Example1: Customization

- Content customization based on browser



## Example2: Factorial

- Print factorial table

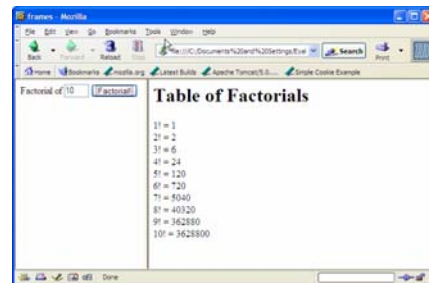


## window Object

- Browser window itself
- Methods
  - open(url,name,[options]) open a new window
  - close()
  - alert() popup alert
  - prompt() user input
- Properties
  - document
  - width, height
  - toolbar, menubar, scrollbar, status, location
  - resizable
  - screenX, screenY

## Example3: Factorial version 2

- Print factorial table on different window



## Regular Expressions

- Compatible with Perl regular expression syntax
- \w Alphanumeric
- \d Numerical digit
- \s White spaces
- . Anything other than newline
- [abcde] Any of a,b,c,d,e
- [a-e] Same as above
- [^a-e] Anything but a to e
- exp?,exp+,exp\* 0 or 1, 1 or more, 0 or more
- exp{x,y} At least x but less than y
- expA | expB expA or expB

## Regular Expressions

- RegExp object

```
var animalSearch = /bear/
var animalSearch = new RegExp("bear")
```
- Methods
  - exec(str) search patten and return result
  - test(str) returns true if match

```
If (animalSearch.test("The little bear")) {
    Alert("I found a bear");
}
```

## Strings and Regular Expressions

- match(RegExpObj)  
Verify input

```
var phone = "416-4403467";
if (phone.match(/d{3}-d{7}/))
    return true;
```
- Replace(RegExpObj, str)  
Replace part of a string

```
var str = "One elephant and two zebras";
var matches = str.replace(/two/, "three");
```

## Example4: Form Validation

